Aftermath: Advances in GPU Crash Debugging

Alex Dunn – Senior Developer Technology Eng.

23 March 2018

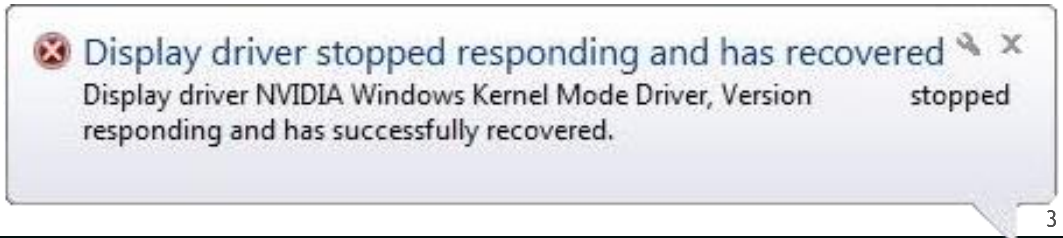
Booth #223 - South Hall

www.nvidia.com/GDC

Why did the GPU crash?

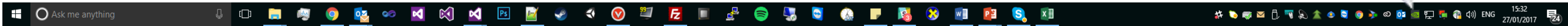
GPU CRASH?

a.k.a. TDR / Hang / Device Removed / Crash



Display driver stopped responding and has recovered

Display driver NVIDIA Windows Kernel Mode Driver, Version stopped responding and has successfully recovered.



WHAT'S HAPPENING?

Behind the scenes…

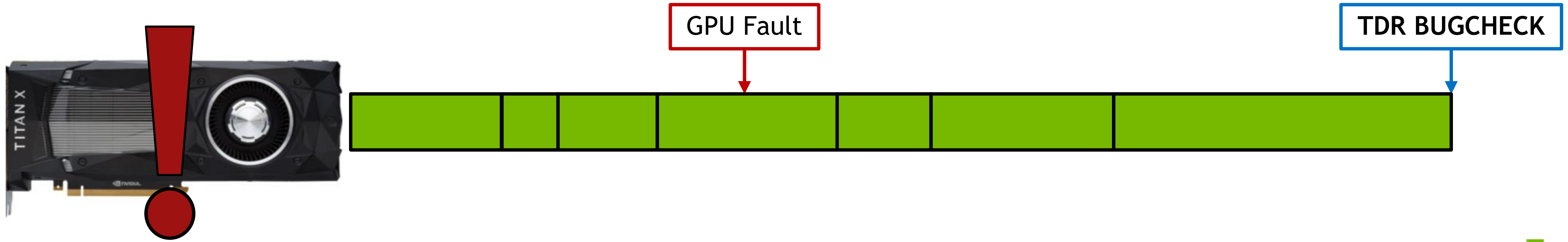
i. OS schedules buffers for execution on GPU

ii. During execution, GPU fault occurs *(or a buffer takes too long to complete)*

iii. GPU scheduler doesn't respond for X seconds *(default is 2s)*

iv. OS raises appropriate bugcheck, KMD attempts to reset engine/adapter

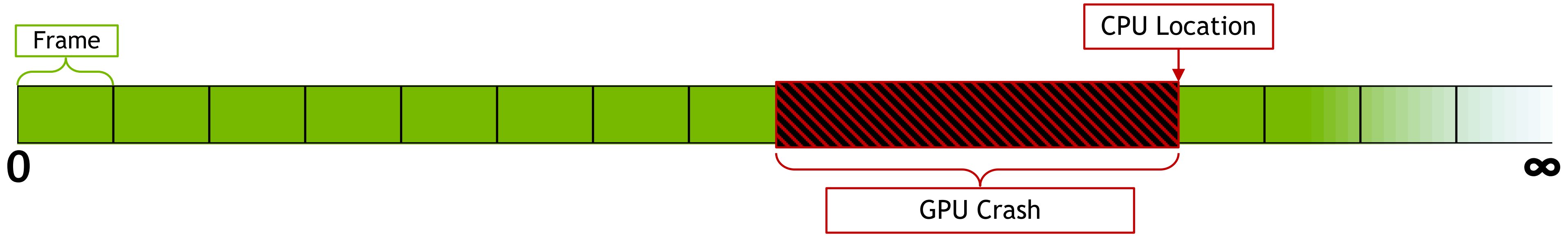
v. Device removed follows… or worse!



DETECTING GPU CRASH

Without Aftermath

- Crash detected based on error code from API (CPU)
- Crash happened sometime in the last N frames of GPU commands…
- CPU call stack of is likely a red-herring



Not useful for debugging!

NVIDIA AFTERMATH (DEBUGGER)

- What is it?
  - Post-mortem GPU debugging tool
  - Helps diagnose GPU crashes (TDRs/Faults)
  - Can be shipped in game – catch crashes “from the wild”
  - Version 2.0 (available soon)

- Support
  - GFX APIs: DX11, DX12 & Vulkan
  - Platforms: GeForce - Windows (and UWP), Linux - (x86, x64)

What does it do?

FEATURE SET

Aftermath 2.0

i. GPU Crash Reason

ii. Page Fault State/Resource Tracking

iii. GPU Checkpoints

iv. ...

GPU CRASH REASON

Two Fundamental Categories

TIME-OUT

i. Driver induced time-out
e.g. unrecoverable fault →

ii. Long running execution
e.g. infinite loop in shader

iii. Incorrect synchronization
e.g. wait without signal

FAULT

i. Page fault
e.g. non-resident read

ii. Invalid page access
e.g. read buffer as texture

iii. Push buffer fault
e.g. malformed commands

iv. Graphics exception
e.g. unaligned CBV

IMPLEMENTATION

After device removed call this:

```
    GFSDK_Aftermath_GetDeviceStatus(
        GFSDK_Aftermath_DeviceStatus* pOutDeviceStatus
    );
```

Possible status:

$$\text{Transition}\left\{\begin{array}{l}\texttt{Unknown}\\ \texttt{Active}\\ \texttt{Stopped}\\ \texttt{Reset}\end{array}\right.$$

$$\text{...}\left\{\begin{array}{l}\texttt{Timeout}\end{array}\right.$$

$$\text{Faults}\left\{\begin{array}{l}\texttt{OutOfMemory}\\ \texttt{PageFault}\\ \texttt{DmaFault}\end{array}\right.$$

RESOURCE TRACKING

KO: Make page faults actionable, by maintaining a table of resources and their VA mapping.

Example:

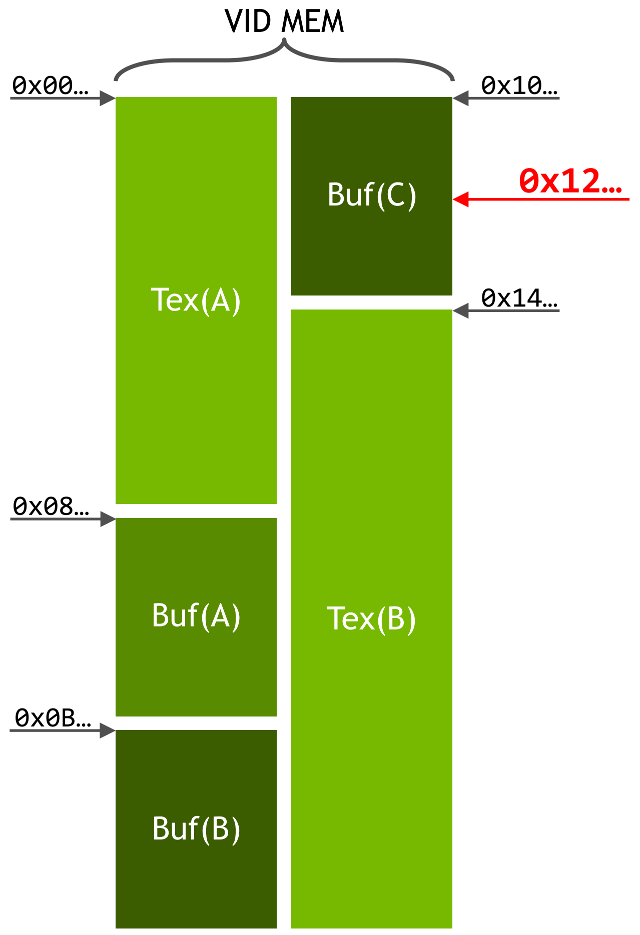
i. Release/Evict ‘Buf(C)’

ii. Access ‘Buf(C)’ in shader

iii. Page fault occurred @ `0x12`

iv. ‘Buf(C)’ last occupied this VA

RESOURCE	BASE VA	SIZE	RELEASED
Tex (A)	0x00	8	0
Buf (A)	0x08	4	0
Buf (B)	0x0B	4	0
Buf (C)	0x10	4	1
Tex (B)	0x14	12	0



IMPLEMENTATION

Once a page fault has occurred and the device removed:

```
    GFSDK_Aftermath_GetPageFaultInformation(
        GFSDK_Aftermath_PageFaultInfo* pOutPageFaultInfo
    );
```

The following can be used to link app and driver resources:

```
    GFSDK_Aftermath_DX12_RegisterResource(
        ID3D12Resource* const pResource,
        GFSDK_Aftermath_ResourceHandle* pOutResourceHandle
    );

    GFSDK_Aftermath_DX12_UnregisterResource(
        GFSDK_Aftermath_ResourceHandle hResource
    );
```

AFTERMATH CHECKPOINTS

(prev. Markers)

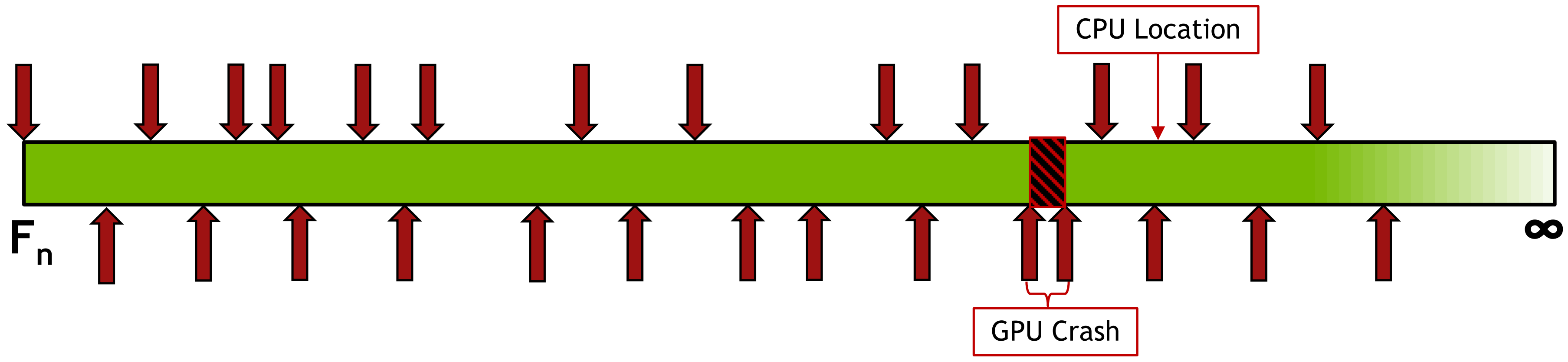
Checkpoints: Narrow in on GPU crash location WRT to command stream

Example:

i. Game inserts user-defined markers in the command stream (CPU)

ii. GPU signals each checkpoint once reached

iii. Last marker reached indicates GPU crash location



IMPLEMENTATION

To inject a checkpoint:

```
    GFSDK_Aftermath_SetCheckpoint(
        GFSDK_Aftermath_ContextHandle hCmdListContext,
        const void* pData, unsigned int size
    );
```

When device removed has been detected:

```
    GFSDK_Aftermath_GetCheckpointData(
        GFSDK_Aftermath_ContextHandle hCmdQueueContext,
        void** outBottomCP, unsigned int* outBottomSize
        void** outTopCP,    unsigned int* outTopSize
    );
```

Vulkan®

- Adding Vulkan support!
- Initially exposing checkpoints as extension →
- Available via the NVIDIA beta developer program:
  - https://developer.nvidia.com/vulkan-driver

```
// VK_NV_device_diagnostic_checkpoints

typedef struct VkCheckpointDataNV {
    VkStructureType sType;
    const void* pNext;
    VkBool32    checkpointTopValid;
    void*       pCheckpointTop;
    VkBool32    checkpointBottomValid;
    void*       pCheckpointBottom;
} VkCheckpointDataNV;


void vkCmdSetCheckpointNV(
    VkCommandBuffer commandBuffer,
    const void* pCheckpointData
);

VkResult vkGetCheckpointDataNV(
    VkQueue queue,
    VkCheckpointDataNV* pCheckpointData
);
```

DirectX® Raytracing (DXR)

- Aftermath supports GPU crash debugging with DXR!
  - All current features supported
- A single ‘DispatchRays’ call can invoke many shaders!
  - Similar problem to ExecuteIndirect
  - Checkpoints aren’t the most helpful…
  - We’re working on improving this for 2.0!

GDC

But what does it do for me???

NO EASY ANSWERS

What does it give me then?

- Not giving you the answer to riddle, it's just a clue!
  - e.g. checkpoints don't tell you which workloads caused a GPU crash.
    - *They tell us what the GPU last finished processing.*
  - e.g. resource tracking doesn't tell us the resource that caused a GPU crash.
    - *It tells us which resources overlap a faulting virtual address.*

GPU CRASH DEBUGGING PROCESS

Some tips! Learned the hard way!!!

i. Collect data on all the crash reports for a given repro!

ii. Find commonality between them (e.g. same shader? shared resources?)

  - Remember, shaders share lot's of code! (*Helpful to look at asm…)*

iii. Divide and conquer the common factors

“CROWD SOURCING”

- Aftermath can be shipped and included in existing crash reporting infrastructure
  - Bucketize crashes by their signature
  - Prioritize fixing more frequent crashes

- Same process applies: confirm an in-house repro using crash signature!

What else is there?

GPU CRASH TOOLBOX?

- More and more options for GPU crash debugging now!
  - PIX support - https://blogs.msdn.microsoft.com/pix/tdr-debugging/
  - DX Debug Layers/GBV improving support
  - ID3D12GraphicsCommandList2::WriteBufferImmediate(…)
  - Aftermath 2.0

- Future:
  - Watch this space, more collaboration and more work still to happen!

QUESTIONS?

Thank you!

```
\0
 |\
 /\
```

Ref.

i. *https://msdn.microsoft.com/en-gb/windows/uwp/gaming/handling-device-lost-scenarios*

ii. *https://msdn.microsoft.com/en-us/library/windows/desktop/bb509553(v=vs.85).aspx*

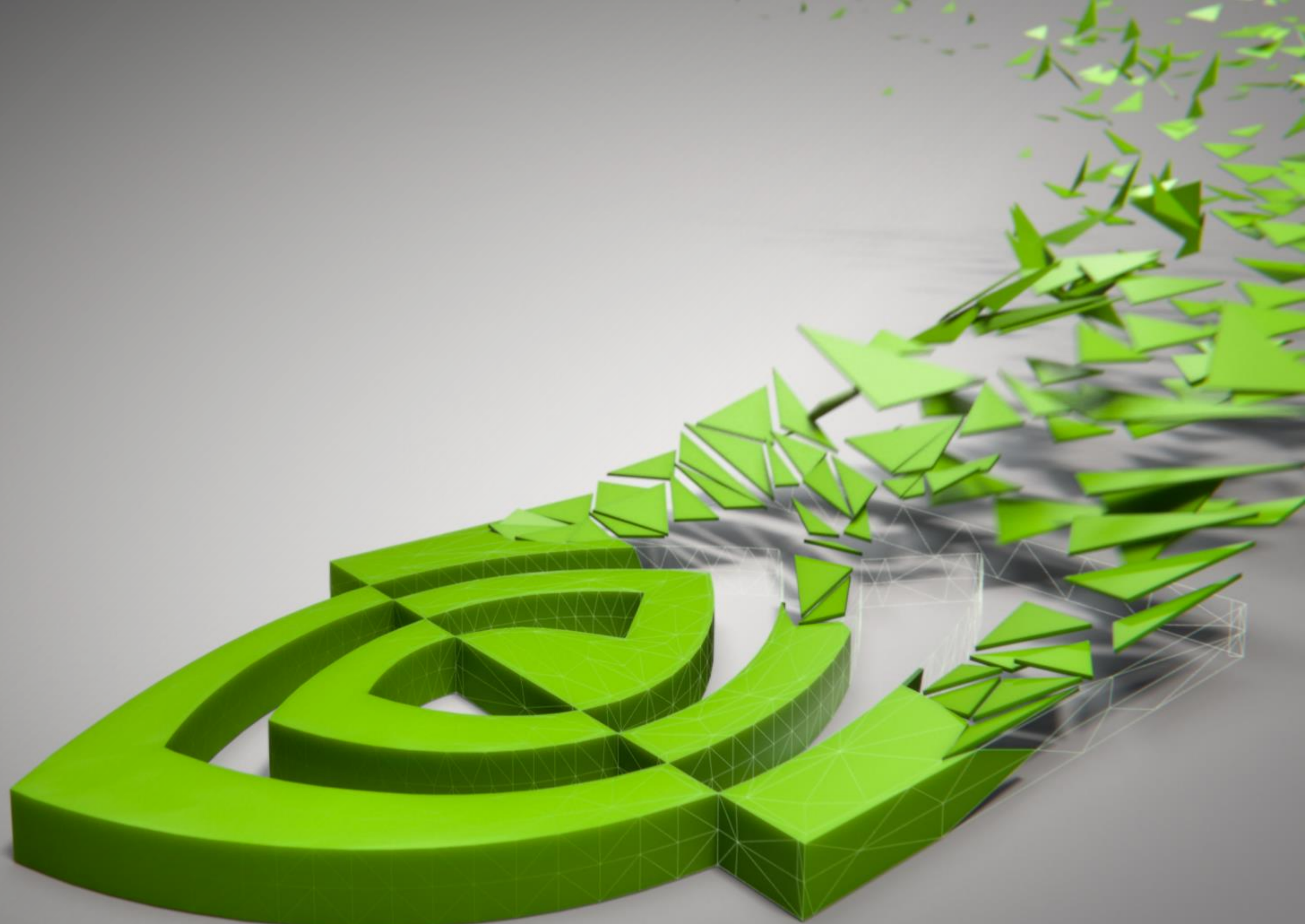
iii. *http://nvidia.custhelp.com/app/answers/detail/a_id/3335/~/tdr-(timeout-detection-and-recovery)-and-collecting-dump-files*

iv. *https://www.khronos.org/registry/vulkan/specs/1.0/html/vkspec.html#devsandqueues-lost-device*

v. *https://developer.nvidia.com/nvidia-aftermath*

vi. https://blogs.msdn.microsoft.com/pix/tdr-debugging

vii. https://developer.nvidia.com/vulkan-driver



Booth #223 - South Hall

www.nvidia.com/GDC